Multi-Voice Audio Playback

Laurel Pardue, Robert McIntyre

April 7, 2010

Micro-Architecture

A quick summary of our system

The audio data ("samples") start in the memory, but are soon pulled into action by the DMA (direct memory access). The DMA sends the samples to a chain of 0 or more *soft-cores*, where they are transformed according to the soft-cores' algorithms. After running the gauntlet of soft-cores, the samples flow first to a buffering FIFO, and finally to a *mixer*, which sends the samples off to be played by speakers of stored in a file.

A more detailed look a our system

Our audio pipeline starts its life with whatever music we want to play already in memory. The music has multiple tracks, or "voices," which we arrange on the memory in contiguous sequences. Therefore, each voice starts at its own start-address, and continues on for as long as it needs. We cap each voice with an end-of-file marker. We make every voice the same length to simplify timing issues, even though that might mean that there is a lot of wasted space in the form of dead areas with no sound.

When the processing starts, the DMA reads in each voice and sends the audio data ("samples") off to a chain of soft-cores. The DMA knows to which chain the samples should be forwarded by looking up the samples' voice in a hash table read at startup. Once the samples enter the soft-core chain, the first soft-core in line takes the samples and performs some sort of audio operation, producing modified audio samples. These samples feed into the next soft-core which performs a different algorithm. As condition for functional correctness, we require the algorithm that each soft-core contains to be able to produce audio samples at the desired 44khz rate, but we allow an arbitrary setup time for the soft-core to get ready. That way, a pipelined processor can take its time and fill up its pipeline completely with no worries. Once the soft-core starts producing samples, it must produce them at a rate of at least 44khz forever. If every soft-core in the chain obeys this timing condition, then the whole chain will as well, so the entire chain of soft-cores can be treated as a single big soft-core

for timing purposes. Our soft-cores are just Lab 5 processors preloaded with specific audio processing algorithms.

Eventually, the samples make their way to the end of the chain of soft-cores. Every chain of soft-cores ends in a FIFO. For this project, we will have 12 separate voices. Therefore, there will be 12 soft-core chains, each with a FIFO at its end, one for each voice. Each of the 12 FIFOs leads to the mixer, which only begins operation once all 12 FIFOs are full. Because of the timing constraints we impose on our soft-cores, once every FIFO has at least one sample, all 12 will forever more be able to supply samples at 44khz. The mixer therefore waits until all 12 FIFOs have elements and then reads samples from all FIFOS at a rate of 44khz. The mixer combines the samples through simple addition and scaling, creating a single stream of sample data, which is sent back from the FPGA to the host computer where we play the stream and/or save it to a file. The mixer ensures that it outputs samples at a rate of 44khz by running everything through its own internal buffer FIFO clocked at 44khz.

Notes

We may implement a system to provide dynamic controls to the soft-cores (such as dynamic volume adjustment or pitch bending) if we have time. Also in the do-if-we-have-time category is a better system for storing sound so that we don't have to waste so much space when a particular voice has nothing to say.

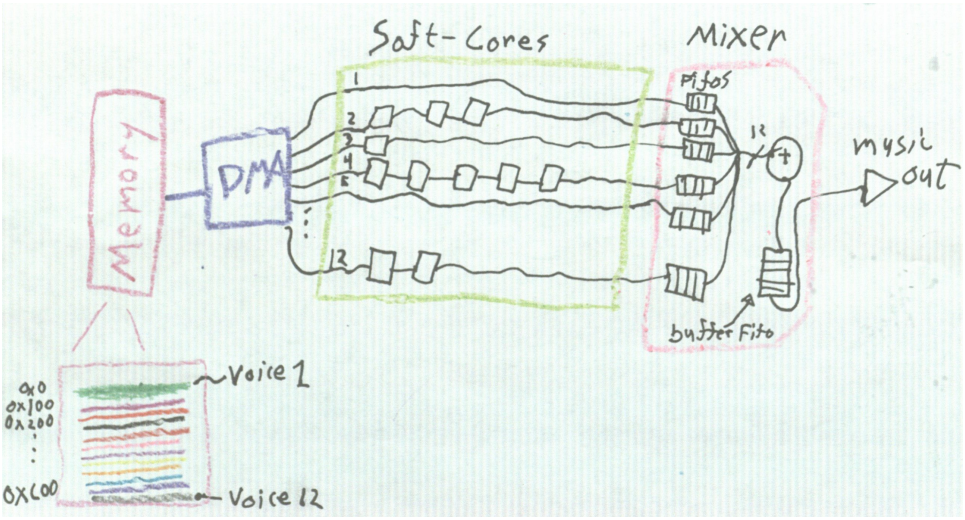

Figure 1: The audio data ("samples") start in the memory, but are soon pulled into action by the DMA (direct memory access). The DMA sends the samples to a chain of 0 or more *soft-cores*, where they are transformed according to the soft-cores' algorithms. After running the gauntlet of soft-cores, the samples flow first to a buffering FIFO, and finally to a *mixer*, which sends the samples off to be played by speakers of stored in a file.